Práctica 2. Test Driven Development

AMPLIACIÓN DE INGENIERÍA DEL SOFTWARE

4.º GII + Matemáticas

Borja Dosuna Hernández

Jaime González Casero

Fecha: 04-05-2021

URJC Escuela Técnica Superior de Ingeniería Informática

ÍNDICE

1. Objetivos

El objetivo de la presente entrega es poner en práctica el **desarrollo guiado por pruebas** mediante una ampliación de la funcionalidad de la Práctica 1. Se ha implementado un servicio Spring que, dada una longitud, divide en líneas de dicha longitud un determinado texto.

Se ha aplicado el TDD con los **14 ejemplos** proporcionados en el enunciado, en orden de aparición, indicando para cada uno de ellos:

- **Test** que prueba el correcto funcionamiento
- **Causa** (en caso de fallo) de que el test no pase
- Código que hace **pasar el test**
- **Refactorización** de ese código

En la refactorización del código se han utilizado **SonarQube** y diversas **funcionalidades proporcionadas por el IDE** SpringSource Tool Suite, destacando entre ellas la funcionalidad “Quick Fix”.

Finalmente, se ha recurrido a la utilización de un repositorio en la plataforma **GitHub** ya creado para la Práctica 1 para facilitar el control de versiones, así como el trabajo colaborativo.

2. Ciclos del TDD

A continuación se detallan los catorce ciclos del proceso llevado a cabo, los cambios se han representado mediante diversas capturas de las **dos nuevas clases** *LineBreakerTest* y *LineBreaker*. Siempre y cuando no se indique lo contrario, todas las capturas de los apartados “Código del test” corresponderán a la clase *LineBreakerTest*; del mismo modo las capturas de “Implementación que hace pasar el test” e “Implementación tras factorizar” corresponden a la clase *LineBreaker*.

Cabe mencionar que, a pesar de que a partir de los primeros tests SonarQube nos recomendaba crear un solo test parametrizado, esta modificación no se realizó hasta la última refactorización de todas, ya que se consideró que la teoría del TDD no aplicaba a la clase *LineBreakerTest.*

Ciclo 1:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string vacio sale un string vacio")
void givenEmptyString_whenLineBreaker_thenReturnEmptyString() {
    //Given
    String input = "";
    //When
    String output = LineBreaker.breakText(input, 2);
    //Then
    String expected = "";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <> but was: <null>
at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenEmptyString_whenLineBreaker_thenReturnEmptyString(LineBreakerTest.java:23)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        return "";
    }
}
```

- Implementación después de refactorizar

No aplica pues no hace falta refactorizar el código.

Ciclo 2:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string que no hace falta partir sale igual")
void givenTestStringAnd4Characters_whenLineBreaker_thenReturnSameString() {
    //Given
    String input = "test";
    //When
    String output = LineBreaker.breakText(input, 4);
    //Then
    String expected = "test";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <test> but was: <>
at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestStringAnd4Characters_whenLineBreaker_thenReturnSameString(LineBreakerTest.jav
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        return text;
    }
}
```

- Implementación después de refactorizar

No aplica pues no hace falta refactorizar el código.

Ciclo 3:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string que no hace falta partir sale igual")
void givenTestStringAnd5Characters_whenLineBreaker_thenReturnSameString() {
    //Given
    String input = "test";
    //When
    String output = LineBreaker.breakText(input, 5);
    //Then
    String expected = "test";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

No aplica pues el pasa el test.

- Implementación que hace pasar el test

No aplica pues es la misma implementación que en el ciclo anterior.

- Implementación después de refactorizar

No aplica pues es la misma implementación que en el ciclo anterior.

Ciclo 4:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con varias palabras sale cortado en dos")
void givenTestWSTestStringAnd4Characters_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test test";
    //When
    String output = LineBreaker.breakText(input, 4);
    //Then
    String expected = "test\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <test
test> but was: <test test>
at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestWSTestStringAnd4Characters_whenLineBreaker_thenReturnBrokenString(LineBreak
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String[] lines = text.split(" ");
            return lines[0]+"\n"+lines[1];
        }
    }
}
```

- Implementación después de refactorizar

```java
@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String[] lines = text.split(" ");
            String output = "";
            for(int i=0;i<lines.length;i++) {
                output += lines[i];
                if(i<lines.length-1)
                    output += "\n";
            }
            return output;
        }
    }
}
```

Ciclo 5:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con dos palabras sale cortado en dos")
void givenTestWSTestStringAnd5Characters_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test test";
    //When
    String output = LineBreaker.breakText(input, 5);
    //Then
    String expected = "test\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

No aplica pues el pasa el test.

- Implementación que hace pasar el test

No aplica pues es la misma implementación que en el ciclo anterior.

- Implementación después de refactorizar

No aplica pues es la misma implementación que en el ciclo anterior.

Ciclo 6:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con dos palabras sale cortado en dos")
void givenTestWSTestStringAnd6Characters_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test test";
    //When
    String output = LineBreaker.breakText(input, 6);
    //Then
    String expected = "test\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

No aplica pues el pasa el test.

- Implementación que hace pasar el test

No aplica pues es la misma implementación que en el ciclo anterior.

- Implementación después de refactorizar

No aplica pues es la misma implementación que en el ciclo anterior.

Ciclo 7:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con cuatro palabras sale cortado en dos")
void givenTestWSTestWSTestWSTestStringAnd9Characters_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test test test test";
    //When
    String output = LineBreaker.breakText(input, 9);
    //Then
    String expected = "test test\ntest test";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <test test
test test> but was: <test
test
test
test>
at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestWSTestStringAnd6Characters_whenLineBreaker_thenReturnBrokenString(LineBrea
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String[] lines = text.split(" ");
            String output = "";
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if(output.charAt(output.length()-1) == ' ') {
                        output = output.substring(0,output.length()-1)+"\n";
                    }
                    else {
                        output += "\n";
                    }
                    output += lines[i];
                    if(i<lines.length-1)
                        output += " ";
                    charCount=0;
                }
                else {
                    output += lines[i];
                    if(i<lines.length-1)
                        output += " ";
                }
            }
            return output;
        }
    }
}
```

- Implementación después de refactorizar

```java
@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String[] lines = text.split(" ");
            String output = "";
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if(output.charAt(output.length()-1) == ' ') {
                        output = output.substring(0,output.length()-1);
                    }
                    output += "\n";
                    charCount=0;
                }
                output += lines[i];
                if(i<lines.length-1)
                    output += " ";
            }
            return output;
        }
    }
}
```

Ciclo 8:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con 2 espacios sale cortado en dos con espacios unicos")
void givenTestStringWithTwoConsecutiveSpacesBetweenWordsAnd4Chars_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test  test";
    //When
    String output = LineBreaker.breakText(input, 4);
    //Then
    String expected = "test\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <test
test> but was: <test
test>
at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestStringWithTwoConsecutiveSpacesBetweenWordsAnd4Chars_whenLineBreaker_thenReturnBrokenString(LineBreakerTest.java:108)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String aux = text.trim().replaceAll(" +", " ");
            String[] lines = aux.split(" ");
            String output = "";
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if(output.charAt(output.length()-1) == ' ') {
                        output = output.substring(0,output.length()-1);
                    }
                    output += "\n";
                    charCount=0;
                }
                output += lines[i];
                if(i<lines.length-1)
                    output += " ";
            }
            return output;
        }
    }
}
```

- Implementación después de refactorizar

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String[] lines = text.trim().replaceAll(" +", " ").split(" ");
            String output = "";
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if(output.charAt(output.length()-1) == ' ') {
                        output = output.substring(0,output.length()-1);
                    }
                    output += "\n";
                    charCount=0;
                }
                output += lines[i];
                if(i<lines.length-1)
                    output += " ";
            }
            return output;
        }
    }
}
```

Ciclo 9:

- Código del test

```
@Test
@DisplayName("Comprobar que cuando entra un string con 3 espacios sale cortado en dos con espacios unicos")
void givenTestStringWithTwoConsecutiveSpacesBetweenWordsAnd6Chars_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test   test";
    //When
    String output = LineBreaker.breakText(input, 6);
    //Then
    String expected = "test\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

No aplica pues el pasa el test.

- Implementación que hace pasar el test

No aplica pues es la misma implementación que en el ciclo anterior.

- Implementación después de refactorizar

No aplica pues es la misma implementación que en el ciclo anterior.

Ciclo 10:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string sin espacios y dos palabras sale cortado en dos")
void givenTestStringWithoutSpacesAndTwoWords_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "testtest";
    //When
    String output = LineBreaker.breakText(input, 5);
    //Then
    String expected = "test-\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
    at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)
    at java.base/java.lang.String.charAt(String.java:712)
    at es.urjc.code.daw.library.book.LineBreaker.breakText(LineBreaker.java:20)
    at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestStringWithoutSpacesAndTwoWords_whenLineBreaker_thenReturnBrokenString(LineBreakerTest.java:129)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {
    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            String[] lines = text.trim().replaceAll(" +", " ").split(" ");
            String output = "";
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (output != "") {
                        if(output.charAt(output.length()-1) == ' ') {
                            output = output.substring(0,output.length()-1);
                        }
                    }
                    else {
                        String aux = lines[i].substring(0, lineLength-1);
                        aux += "-\n";
                        aux += lines[i].substring(lineLength-1);
                        lines[i] = aux;
                    }
                    output += "\n";
                    charCount=0;
                }
                output += lines[i];
                if(i <lines.length-1)
                    output += " ";
            }
            return output.trim();
        }
    }
}
```

- Implementación después de refactorizar

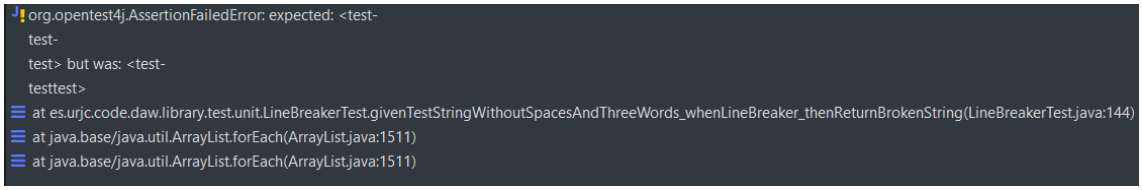
```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
class LineBreaker {

    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            final String[] lines = extracted(text);
            StringBuilder output = new StringBuilder();
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (output.toString().equals("")) {
                        String aux = lines[i].substring(0, lineLength-1);
                        aux += "-\n";
                        aux += lines[i].substring(lineLength-1);
                        lines[i] = aux;
                    } else if(output.charAt(output.length()-1) == ' ') {
                        StringBuilder builderAux = new StringBuilder(output.substring(0,output.length()-1));
                        output = builderAux;
                    }
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                if(i <lines.length-1)
                    output.append(" ");
            }
            return output.toString().trim();
        }
    }

    private static String[] extracted(String text) {
        return text.trim().replaceAll(" +", " ").split(" ");
    }
}
```

Ciclo 11:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string sin espacios y tres palabras sale cortado en tres")
void givenTestStringWithoutSpacesAndThreeWords_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "testtesttest";
    //When
    String output = LineBreaker.breakText(input, 5);
    //Then
    String expected = "test-\ntest-\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test



- Implementación que hace pasar el test

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {

    public static String breakText(String text, int lineLength) {
        if(text.length()<=lineLength) {
            return text;
        }
        else {
            final String[] lines = extracted(text);
            StringBuilder output = new StringBuilder();
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (!output.toString().equals("")) {
                        if(output.charAt(output.length()-1) == ' ') {
                            StringBuilder builderAux = new StringBuilder(output.substring(0,output.length()-1));
                            output = builderAux;
                        }
                    } else {
                        lines[i] = breakOneBigWord(lines[i], lineLength);
                    }
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                if(i <lines.length-1)
                    output.append(" ");
            }
            return output.toString().trim();
        }
    }

    private static String breakOneBigWord(String word, int givenLength) {
        if (word.length()<= givenLength) {
            return word;
        }
        else {
            String aux = word.substring(0, givenLength-1);
            aux += "-\n";
            return aux + breakOneBigWord(word.substring(givenLength-1), givenLength);
        }
    }

    private static String[] extracted(String text) {
        return text.trim().replaceAll(" +", " ").split(" ");
    }
}
```

- Implementación después de refactorizar

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {

    public static String breakText(String text, int lineLength) {
        if(text.length()<= lineLength) {
            return text;
        }
        else {
            final String[] lines = eraseIrregularWhitespaces(text);
            StringBuilder output = new StringBuilder();
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (!output.toString().equals("")) {
                        output = whiteSpaceOnCharAtLineLength(output);
                    } else
                        lines[i] = breakWordsBiggerThanLineLength(lines[i], lineLength);
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                if(i <lines.length-1)
                    output.append(" ");
            }
            return output.toString().trim();
        }
    }

    private static StringBuilder whiteSpaceOnCharAtLineLength(StringBuilder output) {
        if(output.charAt(output.length()-1) == ' ')
            return new StringBuilder(output.substring(0,output.length()-1));
        else
            return output;
    }

    private static String breakWordsBiggerThanLineLength(String word, int givenLength) {
        if (word.length()<= givenLength) {
            return word;
        }
        else {
            String aux = word.substring(0, givenLength-1);
            aux += "-\n";
            return aux + breakWordsBiggerThanLineLength(word.substring(givenLength-1), givenLength);
        }
    }

    private static String[] eraseIrregularWhitespaces(String text) {
        return text.trim().replaceAll(" +", " ").split(" ");
    }
}
```

Ciclo 12:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con palabras mas grandes que la longitud de linea el corte funciona")
void givenTestStringWithWordsLongerThanGivenLength_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test test";
    //When
    String output = LineBreaker.breakText(input, 3);
    //Then
    String expected = "te-\nst\nte-\nst";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <te-
st
te-
st> but was: <te-
st
test>
at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestStringWithWordsLongerThanGivenLength_whenLineBreaker_thenReturnBrokenString(LineBreakerTest.java:156)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {

    public static String breakText(String text, int lineLength) {
        if(text.length()<= lineLength) {
            return text;
        }
        else {
            final String[] lines = eraseIrregularWhitespaces(text);
            StringBuilder output = new StringBuilder();
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (!output.toString().equals(""))
                        output = whiteSpaceOnCharAtLineLength(output);
                    lines[i] = breakWordsBiggerThanLineLength(lines[i], lineLength);
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                if(i <lines.length-1)
                    output.append(" ");
            }
            return output.toString().trim();
        }
    }

    private static StringBuilder whiteSpaceOnCharAtLineLength(StringBuilder output) {
        if(output.charAt(output.length()-1) == ' ')
            return new StringBuilder(output.substring(0,output.length()-1));
        else
            return output;
    }

    private static String breakWordsBiggerThanLineLength(String word, int givenLength) {
        if (word.length()<= givenLength) {
            return word;
        }
        else {
            String aux = word.substring(0, givenLength-1);
            aux += "-\n";
            return aux + breakWordsBiggerThanLineLength(word.substring(givenLength-1), givenLength);
        }
    }

    private static String[] eraseIrregularWhitespaces(String text) {
        return text.trim().replaceAll(" +", " ").split(" ");
    }
}
```

- Implementación después de refactorizar

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {

    public static String breakText(String text, int lineLength) {
        if(text.length()<= lineLength) {
            return text;
        }
        else {
            final String[] lines = eraseIrregularWhitespaces(text);
            StringBuilder output = new StringBuilder();
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (!output.toString().equals(""))
                        output = whiteSpaceOnCharAtLineLength(output);
                    lines[i] = breakWordsBiggerThanLineLength(lines[i], lineLength);
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                if(i <lines.length-1)
                    output.append(" ");
            }
            return output.toString().trim();
        }
    }

    private static StringBuilder whiteSpaceOnCharAtLineLength(StringBuilder output) {
        if(output.charAt(output.length()-1) == ' ')
            return new StringBuilder(output.substring(0,output.length()-1));
        else
            return output;
    }

    private static String breakWordsBiggerThanLineLength(String word, int givenLength) {
        if (word.length()<= givenLength) {
            return word;
        }
        else {
            String aux = word.substring(0, givenLength-1);
            aux += "-\n";
            return aux + breakWordsBiggerThanLineLength(word.substring(givenLength-1), givenLength);
        }
    }

    private static String[] eraseIrregularWhitespaces(String text) {
        return text.trim().replaceAll(" +", " ").split(" ");
    }
}
```

Ciclo 13:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con unas palabras pequeñas y otras grandes el corte funciona")
void givenTestStringWithSomeWordsLongerTAndOthersShorterThanGivenLength_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "test 1234567 test";
    //When
    String output = LineBreaker.breakText(input, 6);
    //Then
    String expected = "test\n12345-\n67\ntest";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

```
org.opentest4j.AssertionFailedError: expected: <test
12345-
67
test> but was: <test
12345-
67 test>
    at es.urjc.code.daw.library.test.unit.LineBreakerTest.givenTestStringWithSomeWordsLongerTAndOthersShorterThanGivenLength_whenLineBreaker_thenReturnBrokenString(LineBreakerTest.java:168)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

- Implementación que hace pasar el test

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {

    public static String breakText(String text, int lineLength) {
        if(text.length()<= lineLength) {
            return text;
        }
        else {
            final String[] lines = removeIrregularWhitespaces(text).split(" ");
            StringBuilder output = new StringBuilder();
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (!output.toString().equals(""))
                        output = whiteSpaceOnCharAtLineLength(output);
                    lines[i] = breakWordsBiggerThanLineLength(lines[i], lineLength, false);
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                output.append(" ");

            }
            return removeIrregularWhitespaces(output.toString().trim());
        }
    }

    private static StringBuilder whiteSpaceOnCharAtLineLength(StringBuilder output) {
        if(output.charAt(output.length()-1) == ' ')
            return new StringBuilder(output.substring(0,output.length()-1));
        else
            return output;
    }

    private static String breakWordsBiggerThanLineLength(String word, int givenLength, boolean checker) {
        if ( (word.length()<= givenLength) && (!checker) )
            return word;
        else if (word.length()<= givenLength)
            return word + "\n";
        else {
            String aux = word.substring(0, givenLength-1);
            aux += "-\n";
            return aux + breakWordsBiggerThanLineLength(word.substring(givenLength-1), givenLength, true) ;
        }
    }

    private static String removeIrregularWhitespaces(String text) {
        return text.trim().replaceAll(" +", " ").replaceAll("\n\n+", "\n").replaceAll("\n ", "\n");
    }

}
```

- Implementación después de refactorizar

```java
package es.urjc.code.daw.library.book;

import org.springframework.stereotype.Service;

@Service
public class LineBreaker {

    private LineBreaker() {
        //not called
    }
    public static String breakLine(String text, int lineLength) {
        if(text.length()<= lineLength) {
            return text;
        }
        else {
            final String[] lines = removeIrregularWhitespaces(text).split(" ");
            StringBuilder output = new StringBuilder("");
            int charCount = 0;
            for(int i=0;i<lines.length;i++) {
                charCount += lines[i].length();
                if(charCount>lineLength) {
                    if (!output.toString().equals(""))
                        output = whiteSpaceOnCharAtLineLength(output);
                    lines[i] = breakWordsBiggerThanLineLength(lines[i], lineLength, false);
                    output.append("\n");
                    charCount=0;
                }
                output.append(lines[i]);
                output.append(" ");

            }
            return removeIrregularWhitespaces(output.toString().trim());
        }
    }

    private static StringBuilder whiteSpaceOnCharAtLineLength(StringBuilder output) {
        if(output.charAt(output.length()-1) == ' ')
            return new StringBuilder(output.substring(0,output.length()-1));
        else
            return output;
    }

    private static String breakWordsBiggerThanLineLength(String word, int givenLength, boolean checker) {
        if ( (word.length() < givenLength) && (!checker) )
            return word;
        else if (word.length() <= givenLength)
            return word + "\n";
        else {
            String aux = word.substring(0, givenLength-1);
            aux += "-\n";
            return aux + breakWordsBiggerThanLineLength(word.substring(givenLength-1), givenLength, true) ;
        }
    }

    private static String removeIrregularWhitespaces(String text) {
        return text.trim().replaceAll(" +", " ").replaceAll("\n+", "\n").replace("\n ", "\n");
    }

}
```

Ciclo 14:

- Código del test

```java
@Test
@DisplayName("Comprobar que cuando entra un string con una palabra mas larga que la longitud el corte funciona")
void givenTestStringWithOnlyOneWordLongerThanGivenLength_whenLineBreaker_thenReturnBrokenString() {
    //Given
    String input = "123456789";
    //When
    String output = LineBreaker.breakText(input, 3);
    //Then
    String expected = "12-\n34-\n56-\n789";
    assertEquals(expected,output);
}
```

- Mensaje de error obtenido al fallar el test

No aplica pues el pasa el test.

- Implementación que hace pasar el test

No aplica pues es la misma implementación que en el ciclo anterior.

- Implementación después de refactorizar

En este caso la captura corresponde a la clase *LineBreakerText.*

```
package es.urjc.code.daw.library.test.unit;


import static org.junit.jupiter.api.Assertions.assertEquals;

@DisplayName("Tests unitarios para el desarrollo guiado por pruebas")
class LineBreakerTest {
    LineBreaker lineBreaker;

    @ParameterizedTest(name = "Test del ejemplo {index}")
    @MethodSource("values")
    void absoluteTest(String input, int lineLength, String output) {

        //When
        String brokenLine = LineBreaker.breakLine(input, lineLength);

        //Then
        assertEquals(brokenLine,output);
    }

    public static Collection<Object[]> values() {

        Object[][] data = {
            {""                    , 2,                         ""},
            {"test"                , 4,                     "test"},
            {"test"                , 5,                     "test"},
            {"test test"           , 4,                "test\ntest"},
            {"test test"           , 5,                "test\ntest"},
            {"test test"           , 6,                "test\ntest"},
            {"test test test test" , 9,      "test test\ntest test"},
            {"test  test"          , 4,                "test\ntest"},
            {"test   test"         , 6,                "test\ntest"},
            {"testtest"            , 5,               "test-\ntest"},
            {"testtesttest"        , 5,        "test-\ntest-\ntest"},
            {"test test"           , 3,          "te-\nst\nte-\nst"},
            {"test 1234567 test"   , 6,    "test\n12345-\n67\ntest"},
            {"123456789"           , 3,        "12-\n34-\n56-\n789"}
        };

        return Arrays.asList(data);
    }

}
```